

## Introduction

The Bug Logging Process Document is developed for internal use in NETMONASTERY to establish and implement a standardized approach for identifying, logging, and managing bugs within the organization. This document serves as a guide to ensure efficient and effective practices for logging bugs in the software. By following this process, teams can ensure timely resolution of bugs and maintain the quality of services delivered to customers

## Bug Logging Process

The Bug Logging Process is an official document that outlines the formal procedure established by NETMONASTERY for identifying, logging, and managing bugs. It provides a comprehensive, step-by-step guide to systematically record and address bug issues in the software. This document includes the scope, input process, process flow, and expected output to ensure effective bug tracking and resolution.

Coding and Implementation

| Name of Process | Bug Logging Process | | |
| --- | --- | --- | --- |
| Process Owner | Product Manager | | |
| Created By | Christy Peedikaparambil | Date | 27/11/2024 |
| Last Updated By |  | Date |  |

| Process Purpose | This document outlines the steps to report an engineering bug in the GitHub crossroads repository ensuring that each report includes the necessary information for timely resolution. | | | |
| --- | --- | --- | --- | --- |

| Process Scope | Members from the following teams can report bugs to the engineering team:   * Support Team * DARC Team * Sales Team * Engineering Team | | |
| --- | --- | --- | --- |

| Process Input | The process begins when a bug is identified by a team member. | | |
| --- | --- | --- | --- |

| Process Flow | 1. Login into GitHub:    1. To report a bug open GitHub and navigate to the **crossroads** repository. 2. Create a New Issue:    1. Click **New Issue** in the repository. 3. Select Issue Type:    1. A list of options will appear. Navigate to the **Engineering Bug** and click **Get Started**. 4. Complete the Required Fields:    1. Bug Title: Enter a title describing the bug.    2. Customer Name: Enter the customer name. If raised internally, enter "Internal."    3. Environment Type: Specify whether the environment is SaaS, On-prem, or Both.    4. Priority: Select the appropriate priority level (P1, P2, P3, or P4).    5. Description: Add a description to describe the bug. 5. Attach Supporting Files:    1. Attach screenshots and other files that would help reproduce or identify the bug. 6. Submit the Issue:    1. Click **Submit New** **Issue** to save the bug in the crossroads repository. 7. Validation and Quality checks:    1. Ensure all the mandatory fields are filled and the attached files are relevant to the bug.    2. Confirm the priority level accurately that reflects the impact of the bug. | | |
| --- | --- | --- | --- |

| Process Output | 1. This process ensures that bug reports are accurately logged and assigned to the default assignee on **Github crossroads** repository. 2. It guarantees clear communication between both the reporter and the reviewer, providing transparency on the bug’s status and expected resolution. | | |
| --- | --- | --- | --- |

| Exceptions to Normal Process Flow | NA | | |
| --- | --- | --- | --- |

| Control Points and Measurements | 1. Must have access to the GitHub crossroads repository and details about the bug including the title, customer name, environment type, and priority level 2. The Product Manager ensures that all the mandatory fields are filled out when reporting a bug. 3. Requests should be properly categorized and include the necessary information to support the request. | | |
| --- | --- | --- | --- |

Version History

| Version | Date | Author | Approved By | Status |
| --- | --- | --- | --- | --- |
| v1.0 | 27/11/2024 | Christy Peedikaparambil | Swati Das Gupta | Draft |